Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Aplikace směrovacích protokolů v simulačním prostředí OMNeT++

Application of Routing Protocols in OMNeT++ Simulation Environment

2014 **Jan Kvapil**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Jan Kvapil**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Aplikace směrovacích protokolů v simulačním prostředí OMNeT++
Application of Routing Protocols in OMNeT++ Simulation Environment

Zásady pro vypracování:

1. Popište všechny známé směrovací protokoly.
2. V simulačním prostředí OMNeT++ vytvořte, nasimulujte a vyhodnoťte vzorové příklady použití směrovacích protokolů RIP, OSPF a BGP.
3. Celý postup zdokumentujte formou zadání laboratorních cvičení do odborného předmětu.

Seznam doporučené odborné literatury:

KLAUS WEHRLE, Mesut Güneş. *Modeling and tools for network simulation.* Online-Ausg. Berlin: Springer, 2010. ISBN 978-364-2123-313.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Libor Michalek, Ph.D.**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA * FAKULTA elektrotechniky a informatiky * OSTRAVA *

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *22. dubna 2014*

.....
podpis studenta

Poděkování

Rád bych poděkoval Ing. Liboru Michalkovi, Ph.D. za odbornou pomoc a konzultaci, které pohotově poskytl, kdykoliv bylo potřeba, při vytváření této bakalářské práce. Dále děkuji své rodině a přátelům, ať už za důvěru, kterou ve mně vložili, nebo i za povzbudivé slovo. A velké poděkování patří i mé nejvzácnější spolubydlící, která ví, kdy člověk potřebuje podpořit, povzbudit a motivovat.

Abstrakt

Cílem této bakalářské práce je popsat všechny známé směrovací protokoly a také vytvořit praktické ukázky aplikace směrovacích protokolů RIP, OSPF a BGP v simulačním prostředí OMNeT++. Celý postup vytváření jednotlivých simulací je zdokumentován a krok po kroku popsán a poslouží jako zadání laboratorních cvičení v odborném předmětu. V této práci je také popsáno prostředí a instalace simulátoru OMNeT++ a také instalace přídavného balíčku INET.

Klíčová slova

RIP; OSPF; BGP; OMNeT++; Simulace;

Abstract

The main objective of this bachelor work is to create simulations of routing protocols RIP, BGP and OSPF in simulator OMNeT++. Creating of simulations are step-by-step described and these descriptions will serve as tasks for laboratories for expert subject. Other objectives of this thesis are to describe all known routing protocols and describe environment and installation of OMNeT++ and framework INET.

Key words

RIP; OSPF; BGP; OMNeT++; Simulation;

Seznam použitých zkratek

Zkratka	Význam
AFI	Address Family Identifier
ARPANET	Advanced Research Projects Agency Network
AS	Autonomní systém
BGP	Border Gateway Protocol
CIDR	Classless Inter-Domain Routing
CPU	Central processing unit
EIGRP	Enhanced Interior Gateway Routing Protocol
GNU GPL	GNU General Public License
IDE	integrated development environment
IETF	Internet Engineering Task Force
IGRP	Interior Gateway Routing Protocol
IP	Internet Protocol
IS-IS	Intermediate System to Intermediate System
LSA	Link-State Advertisement
NED	Network DEscription
OSPF	Open Shortest Path First
RFC	Request For Comments
RIP	Routing Information Protocol
RIPng	RIP next generation
TCP	Transmission Control Protocol
TTL	Time-To-Live
UDP	User Datagram Protocol
VLSM	Variable Length Subnet Masking

Obsah

Úvod

Už na počátku počítačových sítí si vědci uvědomovali, že by bylo přínosné, kdyby dokázali jednotlivé prvky mezi sebou automaticky komunikovat a vyhledávat nejlepší cesty, po kterých by následně posílaly data z jednoho místa v síti do druhého.

Tento proces se nazývá směrování a slouží pro určení nejlepší cesty, kterou se pošlou data ke svému cíli. Proces zajišťují zařízení nazvané směrovače a používají k tomu směrovací protokoly.

První směrovací protokoly vznikaly už v dobách ARPANETu. Jak se však sítě postupně zvětšovaly, byly kladeny vyšší a vyšší nároky na směrování a taky na samotné směrovací protokoly. Proto za dobu své existence prošly tyto protokoly velkým vývojem, kdy v podstatě s každou novou verzí reagovaly na nějaký technologický pokrok, ať už to bylo zvyšování počtu prvků v síti, nebo v poslední době třeba příchod protokolu IPv6. Dnes díky toho existuje velké množství protokolů s odlišnými vlastnostmi a vhodností použití pro určitý typ sítě.

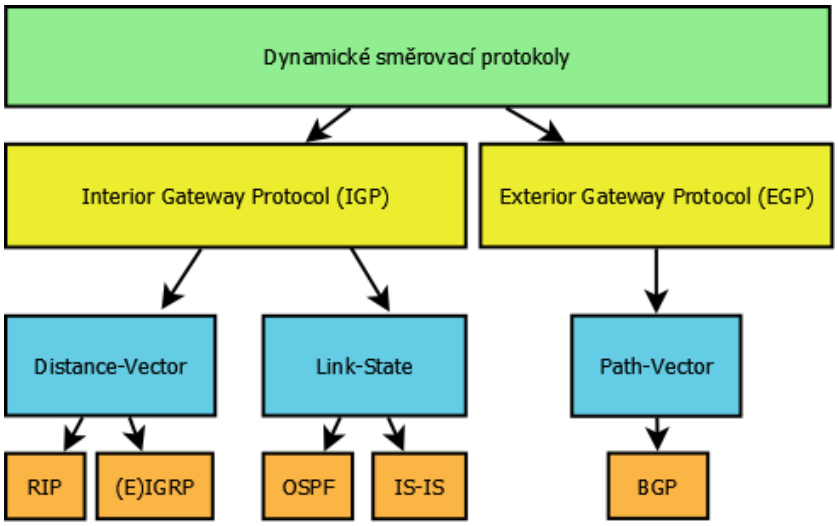
Tato práce si dává za úkol popsat ty nejpoužívanější směrovací protokoly, jako jsou například EIGRP, BGP nebo OSPF. A následně simulovat jejich vlastnosti v simulační prostředí OMNeT++. Simulační prostředí jako takové jsou velmi vhodnými nástroji pro určení vhodnosti různých technologií bez nutnosti pořizovat reálné síťové prvky za velké finanční prostředky.

Samotné simulace budou zpracovány formou zadání laboratorních cvičení odborného předmětu.

1 Popis všech známých směrovacích protokolů

Směrovací protokoly byly vytvořeny za účelem směrování dat v síti. Samotný proces směrování má za úkol doručit (směrovat) data ze zdroje do cíle. V jednoduchých, malých sítích lze tohoto dosáhnout jednoduchý a celkem efektivním způsobem – statickým směrováním. Ve statickém směrování je cesta k cíli manuálně nakonfigurována správcem sítě. Jak však postupem času počítačové sítě rostly a připojovalo se do nich stále více a více uživatelů, stávalo se statické směrování stále více neefektivním. A z tohoto důvodu vznikly Směrovací protokoly a tzv. dynamické směrování. Ty mají za úkol automaticky vypočítat a zvolit nejvhodnější cestu, po které jsou následně poslány data směrem k jejich cíli. V dnešní době existuje několik směrovacích protokolů, které se mohou lišit algoritmem pro výpočet cest, nebo podporou různých síťových technologií, jako například IPv4/IPv6, nebo dalšími různými vlastnostmi. Primární zařízení, které obstarává směrování je Směrovač (Router). V dnešní době ale existují další zařízení podporující směrování, např. přepínače pracující na 3. vrstvě TCP/IP modelu (L3 Switch).

1.1 Rozdělení směrovacích protokolů



Obrázek 1.1: *Rozdělení směrovacích protokolů [3]*

Směrovací protokoly se mohou dělit podle mnoha kritérií. Například to mohou být (Obrázek 1.1):

Podle oblasti použití

- Interior Gateway Protocol – použití uvnitř Autonomních Systémů (AS)
- Exterior Gateway Protocol – použití pro směrování mezi AS

Podle naplňování směrovací tabulky

- Distance-Vector – použití vektorů vzdáleností do vzdálených sítí
- Link-State – zařízení drží informaci o celé topologii sítě a stavech jednotlivých linek

Podle dalších vlastností

- Podpora VLSM/CIDR
- Podpora IPv6
- Rychlost konvergence
- …

1.2 Protokoly založené na vektoru vzdálenosti (Distance-Vector)

Směrovací protokoly používající vektor vzdálenosti (Distance-Vector) v pravidelných intervalech rozesílají svou vlastní směrovací tabulku svým bezprostředním sousedům. Ti si poté k záznamům z přijaté tabulky přičtou svou vlastní vzdálenost – svůj vlastní vektor vzdálenosti - a pošlou aktualizovanou tabulku svým bezprostředním sousedům. Současně každý směrovač aktualizuje svou směrovací tabulku. Ve chvíli, kdy každý směrovač v síti přijal všechny aktualizace a má aktualizovanou vlastní směrovací tabulku, dochází ke konvergenci sítě. Každý ze směrovačů má tak záznamy o vzdálenosti ostatních směrovačů. Ostatní parametry sítě však směrovače neznají, tzn., nemají konkrétní informace o ostatních směrovačích, ani o topologii sítě.

Výhodou protokolů s vektorem vzdáleností je menší náročnost na CPU a na šířku pásma.

Oproti tomu nevýhodou je pomalejší konvergence, než v případě protokolů se Stavem Linky a to díky faktu, že se změny v síti rozesílají pouze mezi sousedy, neaktualizuje se celá síť najednou. Proto jsou starší protokoly s vektorem vzdáleností ne příliš vhodné pro rozsáhlé WAN sítě. Dalším problémem je tzv. Počítání do nekonečna. V síti mohou vzniknout smyčky. Tomuto problému se předchází několika způsoby – pomocí TTL (Time-To-Live, definování maximální možné vzdálenosti), rozdělení horizontu (Split Horizont, informace se neposílají na rozhraní, odkud přišly) a další. [2][3]

Pro výpočet cesty se používá několik směrovacích algoritmů Bellman-Ford (RIP), nebo třeba DUAL FSM (EIGRP).[1]

1.2.1 RIPv1 - Routing Information Protocol version 1

Protokol RIPv1 je jedním z nejstarších směrovacích protokolů. Vychází z algoritmů, které už v 50. letech 20. století popsali vědci R. Bellman, L. Ford a D. Fulkerson. Dnes je známe po názvy Bellman-Fordův algoritmus nebo algoritmus Ford-Fulkerson. Oba dva se později staly základem směrování v malých sítích a umožnily vznik mnoha směrovacím protokolům, které sice vycházely ze stejných algoritmů, ale v mnoha věcech se lišily a tak

nebyly schopny navzájem spolupracovat. Změnu přineslo až sdružení IETF (Internet Engineering Task Force), které v červnu roku 1988 vydalo dokument RFC 1058 (Request For Comments). Tento dokument popisoval novou a otevřenou podobu směrovacího protokolu používající vektor vzdáleností. Protokol byl pojmenován Routing Information Protocol (RIP).

Formát paketů RIP

Paket protokolu RIP má několik částí (Obrázek 1.2): [1]

- Pole příkazu
- Pole s číslem verze
- Nulové pole
- Pole s identifikátorem rodiny adres (Address Family Identifier, AFI)
- Nulové pole
- Pole s adresou internetové sítě
- Nulové pole
- Nulové pole
- Pole s metrikou

1 oktet	1 oktet	2 oktety	2 oktety	2 oktety	4 oktety	4 oktety	4 oktety	4 oktety
Pole příkazu	Pole s číslem verze	Nulové pole	Pole AFI	Nulové pole	Pole s IP adresou	Nulové pole	Nulové pole	Pole s metrikou

Obrázek 1.2: *Struktura RIP paketu[1]*

Pole AFI, pole s IP adresou a pole s metrikou se mohou v paketu až 25krát. Z toho vyplývá, že je možné poslat až 25 aktualizací položek ze směrovací tabulky. Opakující se pole se pouze připojí za základní paket z obrázku 1.2.

Výpočet vektoru vzdáleností

Metrika protokolu RIP, definovaná podle RFC 1058, je počet přeskoků. Implicitně je nastavena pro každou linku na hodnotu 1. Avšak pomocí mechanismů, také definovaných v RFC 1058, může administrátor směrovače cenu linky změnit manuálně.

Samotné počítání ceny pak probíhá následovně: Směrovač pošle všem bezprostředně sousedícím směrovačům svou směrovací tabulku, kde jsou obsaženy informace o vzdálenosti do známých cílů. Sousední směrovače si uloží směrovací tabulku, přičtou svou vlastní vzdálenost do cíle a rozešlou ji dále. Jak už bylo napsáno dříve, aktualizace se posílají periodicky – u protokolu RIP jednou za 30 sekund.

Jestliže má nějaká cesta vzdálenost větší jak 15, je považována za neplatnou a směrovač posílá svým bezprostředním sousedům o tomto faktu aktualizaci.[1]

Udržování aktuálnosti směrovací tabulky

Další důležitou součástí RIP jsou tři časovače, pomocí kterých se udržuje směrovací tabulka: [1]

- **Aktualizační časovač** – implicitně nastaven na 30 sekund. Udává, před jakou dobou směrovač poslal poslední aktualizaci. Po vypršení pošle aktualizaci a resetuje časovač.
- **Časový limit platnosti cesty** – implicitně nastaven na 180 sekund. Udává platnost dané cesty. Po vypršení je cesta prohlášena za neplatnou.
- **Časovač vyprázdnění cest** – implicitně nastaven na 90 sekund. Spouští se po vypršení platnosti cesty. Po vypršení časovače vyprázdnění cest se daná cesta vymaže ze směrovací tabulky.

Charakteristické rysy protokolu RIP

- Classfull protokol – v aktualizacích se nepoužívají masky podsítí. Podporuje pouze IP adresy rozdělené podle tříd (A, B, C, …). Z toho vyplývá, že protokol nepodporuje CIDR/VLSM
- Jednoduchý protokol, snadná konfigurace
- Pomalá konvergence
- Možnost zahlcení sítě aktualizacemi
- Maximální počet přeskoků je 15
- Všesměrové rozesílání aktualizací

1.2.2 RIPv2

Routing Information Protocol version 2 je upravený a o pár funkcí obohacený RIPv1. Byl definován v několika dokumentech a to například RFC 1388 v roce 1993 anebo naposled v roce 1998 v RFC 2453 Mnoho vlastností mají tedy tyto dva protokoly společných. Odlišné znaky RIPv2:

- Classless protokol – obsahuje masky podsítí v aktualizacích. Z toho vyplývá podpora CIDR/VLSM
- Podpora autentizace
- Vícesměrové aktualizace (na IP adrese 224.0.0.9) [5]

Ke změně došlo i u struktury paketu – bylo například přidáno pole s maskou podsítě. Struktura paketu je zobrazena na obrázku 1.3.

1 oktet	1 oktet	2 oktety	2 oktety	2 oktety	4 oktety	4 oktety	4 oktety	4 oktety
Pole příkazu	Pole s číslem verze	Nulové pole	Pole AFI	Pole se značkou cesty	Pole s IP adresou	Pole s maskou podsítě	Pole s nejbližším přeskokem	Pole s metrikou

Obrázek 1.3: *Struktura paketu RIPv2[1]*

1.2.3 RIPng

RIPng neboli RIP next generation je odpověď na vývoj IP protokolu, konkrétně na IPv6. RIPng přidává oproti svým předchůdcům podporu IPv6. Jeho přesná definice je obsažena v dokumentu RFC 2080 v roce 1997. Další odlišností je např. použití autentizace přes IPsec.[4]

1.2.4 IGRP – Interior Gateway Routing Protocol

Jak se postupem času IP sítě rozpínaly a zvětšovaly se, začali se více a více projevovat nedostatky protokolu RIP. Hlavně jeho použití ve velkých sítích se stával problémem, kdy mohlo docházet k zahlcování sítě díky aktualizacím, nebo omezený počet přeskoků. Špatnou vlastností se také ukázalo to, že RIP protokoly měly pro každý cíl pouze jednu cestu, což byl problém jednak v případě poruchy na cestě, ale rovněž se nerovnoměrně využívali redundantní linky. S odpovědí na tyto a další nedostatky přišla společnost Cisco, která v 80. letech 20. století vyvinula protokol IGRP. Ten odstraňoval zásadní nedostatky RIPu a umožnil tak jeho nasazení ve větších sítích.[1]

Charakteristické vlastnosti IGRP

- Pro středně velké až rozsáhlé AS
- Pro složité topologie
- Dynamické vyrovnávání zátěže
- Několik metrik pro výpočet cesty
- Možnost několika cest do jednoho cíle
- Cisco proprietární protokol
- Classfull – bez podpory CIDR/VLSM

Metriky

Protokol IGRP pro výpočet cesty nepoužívá pouze jednu jedinou metriku jako RIP, ale výpočet provádí pomocí několika metrik. Výpočet probíhá podle určitého vzorce, který mj. zohledňuje také váhy jednotlivých metrik. Vypočtený vektor vzdálenosti se pak může pohybovat v rozmezí 1 až 16 777 215 a nižší hodnoty znamenají lepší cesty. Jsou to: Počet přeskoků, Velikost paketů, Šířka pásma linky, Zpoždění, Zatížení linky, Spolehlivost.

Počet přeskoků je používán pouze k zamezení vytvoření smyček v síti. Pro samotný výpočet nejlepší cesty se nepoužívá. Oproti protokolu RIP je maximální počet přeskoků stanoven na 100, manuálně se dá nastavit až 255.

Velikost paketů udává maximální velikost přenášené jednotky – maximální velikost diagramů. Pro výpočet cest se nepoužívá.

Šířka pásma udává rychlost spojení mezi zařízeními. Implicitně je nastavena na 1,544 Mb/s. Je vhodné přenastavit každou linku na její opravdovou hodnotu, aby byla metrika efektivně použita při výpočtu cest.

Zpoždění udává přibližnou dobu, za kterou se překoná určitá linka. Celkové zpoždění je pak dáno součtem zpoždění jednotlivých linek. Pro každý typ přenosového média používá IGRP průměrnou odezvu daného média.

Zatížení linky je aktuální stavem linky – jak moc je určitá linky zrovna používána, jak moc je využita její kapacita. Může dojít k případům, kdy protokol chybně vyhodnotí vysoké využití linky a přesměruje provoz jinou cestou. Proto má tato metrika při výpočtu cest nízkou váhu.

Spolehlivost představuje množství chyb, ke kterým na médiu dojde. Hodnoty jsou z intervalu 1 – 255 (255 = nejspolehlivější).[1]

Udržování aktuálnosti cest

Podobně jako u protokolu RIP, i u IGRP se používají časovače pro udržení aktuálnosti cesty. Každý ze směrovačů rozesílá v pravidelných intervalech kopii své směrovací tabulky. Směrovač, který aktualizaci přijal, automaticky nahrazuje staré záznamy za záznamy nové. IGRP pracuje se 4 časovači: [1]

- Aktualizační časovač – po uplynutí určitého času (implicitně 90 sekund) se rozesílají aktualizace bezprostředně přilehlým sousedům
- Časovač odstavení cesty – zablokuje aktualizace pro danou cestu. Implicitně trojnásobek aktualizačního časovače + 10 sekund
- Časovač neplatnosti cesty – po uplynutí prohlásí cestu za neplatnou. Resetuje se s přijatou aktualizací určité cesty. Implicitně nastaveno na trojnásobek Aktualizačního časovače
- Časovač vyprázdnění cesty – po uplynutí se neplatná cesta odstraní z tabulky. Implicitně sedmínásobek aktualizační periody

1.2.5 EIGRP – Enhanced IGRP

EIGRP sice vychází částečně ze svého předchůdce IGRP, ale v mnoha ohledech se velmi liší. Jako IGRP, je i EIGRP založen na vektoru vzdálenosti, na druhou stranu má i některé vlastnosti typické pro směrování se stavem linky. V podstatě spojuje to nejlepší z obou typů

směrování. I proto je v některých literaturách zařazován do skupiny Hybridních směrovacích protokolů.

Jednou ze zásadních změn je použití nového směrovacího algoritmu – DUAL. Ten umožňuje odhalování smyček v síti a hledat alternativní cesty ještě před přijetím aktualizace. Toto urychluje konvergenci bez zvýšení rizika vytvoření smyček.

Základní charakteristické rysy:

- Proprietární Cisco protokol
- Classless - podpora CIDR/VLSM
- Algoritmus DUAL (difúzní aktualizační algoritmus)
- Rychlá konvergence
- Kompatibilní s IGRP

Zásadní vylepšení v protokolu

Společnost Cisco zavedla v protokolu EIGRP několik podstatných změn, kterými reagovala na trendy a vývoj počítačových sítí. Jednou z důležitých vlastností je třeba podpora masek podsítí nebo beztřídního směrování (VLSM/CIDR). Významný pokrok byl také učiněn ve využití šířky pásma. Při konvergenci se neposílá celá směrovací tabulka, ale pouze její část, která byla změněna. Navíc odesílání aktualizací probíhá pouze v případě změny topologie, neposílají se v pravidelných intervalech. Díky toho se snížila režie i v konvergentní síti – posílají se pouze malé pakety HELLO, které tolik nezahlcují síť. HELLO paket má za úkol zjišťovat v síti nově připojené zařízení, odhalovat nedostupné zařízení, nebo také identifikovat nedostupné cesty. Směrovače mezi sebou posílají pakety HELLO v pravidelných intervalech. Jestliže přestanou přicházet do směrovače pakety z druhého směrovače, znamená to, že druhý směrovač již není aktivní. [1]

Tabulky

Novinkou v EIGRP je také přidání několika dalších tabulek. Každá z nich ukládá informace o určité části sítě. Nejdůležitější tabulky jsou: [1]

- **Tabulka sousedů** – obsahuje vztahy o jednotlivých přilehlých sousedních směrovačích
- **Směrovací tabulka** – obsahuje cesty ke známým cílům. Může obsahovat až 6 různých cest pro jeden cíl.
- **Tabulka síťové topologie** – informace potřebné pro výpočet vzdálenosti cíle. Jsou zde uloženy pole jako Šířka pásma, Celkové zpoždění, Zdroj cesty (číslo směrovače, který tuto cestu oznámil), a další.

1.3 Protokoly založené na stavu linky (Link-State)

Protokoly založené na stavu linky (někdy také nazývané jako protokoly nejkratší cesty) byly vyvinuty na přelomu 80. a 90. let minulého století. Na rozdíl od protokolů, které používají vektor vzdálenosti, mají protokoly Link-State kompletní přehled o topologii sítě – drží si informace o jednotlivých směrovačích a jejich propojení v síti. Po síti pak každý směrovač rozesílá své vlastní LSA (Link-State Advertisement), což jsou informace o stavu linky. Z těchto LSA si pak každý směrovač konstruuje vlastní mapu sítě. Zároveň si podle LSA vypočítává vzdálenosti k jednotlivým cílům, porovná je se směrovací tabulkou a případně tabulku aktualizuje.

Důležitým faktem je, že rozesílání LSA se spouští pouze při nějaké události (např. změna v síti), to znamená, že oproti protokolům RIP nebo IGRP se LSA zprávy neposílají periodicky. To snižuje režii v síti a také výrazně urychluje konvergenci.

Stejně jako protokoly s vektorem vzdáleností, i protokoly se stavem linky mají své nevýhody. Jednou z nich je možné zahlcení sítě při počátečním rozesílání LSA zpráv. Další nedokonalostí je vysoká náročnost na paměť a výpočetní výkon, protože se počítají a udržují informace o celé síti. [1][2]

1.3.1 OSPF – Open Shortest Path First

Na konci 80. let se začaly projevovat problémy směrování s použitím vektoru vzdálenosti. S odpovědí přišlo sdružení IETF, které vydalo protokol OSPF. Poprvé byl definován v dokumentu RFC 1131, který po krátké době byl nahrazen RFC 1247. Nová specifikace obsahovala tolik změn, že protokol by označen jako OSPFv2. Poslední aktualizace je obsažena v dokumentu RFC 2328 pro OSPFv2 a v roce 2008 byl vydán dokument RFC 5340, který definuje OSPFv3 – tato verze je určena pro IPv6 protokol.

Charakteristické rysy OSPF

- Podpora VLSM/CIDR
- Metrikou je cena linky (celková cena je pak dána součtem propustnosti všech linek cesty)
- Podporuje rozdělení zátěže mezi linky se stejnou metrikou (Load Balancing)
- Dijkstrův algoritmus
- Rozdělení síti na oblasti

Metrika

Metrikou je cena linky. Je to bezrozměrná veličina a je nutné ji nastavit manuálně. Nastavuje se na každé rozhraní směrovače zvlášť a nižší hodnota je lepší. [1]

Oblasti

OSPF umožňuje síť rozdělit na menší části, tzv. oblasti, díky kterým je pak síť lépe škálovatelná a rychleji konverguje. Aktualizace se pak můžou rozesílat jak uvnitř oblastí, tak i

mezi oblastmi samotnými. Vždy musí existovat Oblast 0, tzv. páteřní oblast, která spojuji všechny ostatní oblasti. Páteřní oblast s ostatními oblastmi spojují tzv. hraniční směrovače. [1]

Datová struktura OSPF zpráv

OSPF používá pět různých typů zpráv. Každý typ je určen pro jinou operaci. Jsou to pakety: [1]

- HELLO
- S požadavkem na stav linky
- S popisem databáze
- S aktualizací stavu linky (LSA)
- S potvrzením stavu linky

V hlavičce paketu jsou dále obsaženy pole: [1]

- Číslo verze
- Typ
- Délka paket
- ID směrovače
- ID oblasti
- Kontrolní součet
- Typ autentizace
- Autentizace

Jako výhody OSPF by se dala označit jeho otevřenost – není závislý na výrobci. Velkou výhodou je jistě taky rychlá konvergence. Na druhou stranu je oproti protokolu RIP celkem složitý a náročný na výkon síťových zařízení. [1][2]

1.3.2 IS-IS – Intermediate System to Intermediate System

Další zástupcem protokolů se stavem linky je protokol IS-IS. Původně byl vytvořen v roce 1980 jako ISO norma. Autorem je společnost DEC. To znamená, že zpočátku nebyl tento protokol otevřený. Do IS-IS byly později přidány nové funkce – směrování paketů v IP sítích, a následně byl i zveřejněn v RFC 1195. Je využíván hlavně velkými ISP. IS-IS je založený na Dijkstrově algoritmu a velmi se podobá protokolu OSPF. Při porovnání s OSPF se považuje za stabilnější. Jako metriku používá cenu cesty. V protokolu IS-IS jsou tři typy oblastí: [2][7]**Chyba! Nenalezen zdroj odkazů.**

- Úroveň 1 – intra
- Úroveň 2 – inter
- Mezilehlá úroveň (úroveň 1-2) – intra/inter – dovoluje komunikaci mezi různými úrovněmi

1.4 Protokoly s vektorem cesty (Path-Vector)

Protokoly s vektorem trasy vychází z protokolů s vektorem vzdáleností – používá také vektor vzdálenosti cíle a k tomu přidává znalost úplné cesty do cíle. Díky tomu se dají odhalit smyčky snadněji než u směrování s vektorem vzdálenosti.

1.4.1 BGP – Border Gateway Protocol

BGP protokol je určen pro směrování mezi autonomními systémy. Od svého vzniku v roce 1994 prošel mnoha změnami. Aktuální je verze BGP4, ostatní jsou dnes již zastaralé. BGP4 je v podstatě jediný vnější protokol používaný v IP sítích. Používá spolehlivý protokol TCP pro přenos dat. Tvoří jádro směrování v internetu. [2][6][7]

Zprávy BGP [6]

Protokol BGP používá 4 druhy zpráv pro komunikaci se svými sousedy. Jsou to:

- OPEN – počátek komunikace
- UPDATE – propagování/odstranění cest
- NOTIFICATION – oznamuje chybu. Po vyslání zprávy dojde k přerušení spojení se sousedním směrovačem
- KEEPALIVE – udržování spojení

1.5 Porovnání vlastností protokolů

Souhrn jednotlivých charakteristických rysů je uveden na Obrázku 1.4 a na Obrázku 1.5 jsou pak zobrazeny vlastnosti algoritmů směrovacích protokolů.

Protokol	Algoritmus	Otevřený	Vnitřní/vnější	Aktualizace	Metrika	Podpora VLSM/CIDR	Sumarizace
RIPv1	Vektor vzdálenosti	ano	vnitřní	30s	počet skoků	ne	automatická
RIPv2	Vektor vzdálenosti	ano	vnitřní	30s	počet skoků	ano	automatická
IGRP	Vektor vzdálenosti	ne	vnitřní	90s	složená	ne	automatická
EIGRP	DUAL	ne	vnitřní	na základě změny	složená	ano	automatická /manuální
OSPF	stav linky	ano	vnitřní	na základě změny	cena	ano	manuální
IS-IS	stav linky	ano	vnitřní	na základě změny	cena	ano	automatická
BGP	vektor cest	ano	vnější	na základě změny		ano	automatická

Obrázek 1.4: *Porovnání základních rysů protokolů [2]*

	Vektor vzdáleností	DUAL	Stavů linek	Vektor cest
škálovatelnost	malá	vysoká	dobrá	vynikající
zátěž sítě	vysoká	malá	malá	malá
zátěž paměti	malá	střední	vysoká	vysoká
zátěž CPU	malá	malá	vysoká	střední
konvergence	pomalá	rychlá	rychlá	střední
konfigurace	snadná	snadná	středně složitá	složitá

Obrázek 1.5: *vlastnosti algoritmů[2]*

2 Simulátor OMNeT++ a frameworky

2.1 Simulační prostředí OMNeT++

2.1.1 Popis

OMNeT++ je simulační nástroj založený na C++, je šířen pod GNU GPL licencí – pro nekomerční účely je zcela zdarma a samotné vývojové prostředí (IDE) je založené na platformě Eclipse. OMNeT++ je primárně určen k simulaci komunikačních sítí, ale díky jeho flexibilní architektuře se dá využít i v mnoha jiných odvětvích.

2.1.2 Instalace

Popis instalace je pro verzi prostředí OMNeT++ 4.4.1 pro systém Windows. Postup instalace starších verzí nebo verzí pro jiné operační systémy se může lišit.

(Pozn. Kompletní návod na instalaci můžete nalézt na [9])

1. Na oficiálních stránkách projektu `http://omnetpp.org` stáhneme nejnovější verzi simulačního prostředí OMNeT++ pro systémy Windows
2. Umístíme a rozbalíme stažený soubor `omnetpp-<ver>-src-windows.zip` (kde <ver> určuje verzi simulátoru) tam, kde chceme OMNeT++ nainstalovat. Cesta k rozbalenému archivu by neměla obsahovat mezery (např. c:/omnetpp-4.4.1/)
3. Přesuneme se do nově rozbalené složky (např. omnetpp-4.4.1), která obsahuje složky images, mys, aj. a soubory mingwenv.cmd, configure, a další.
4. Spustíme soubor mingwenv.cmd – otevře se okno programu MINGW32, určené pro konfiguraci prostředí OMNeT++
5. Zadáme příkaz `./configure` a poté příkaz `make` – proběhne konfigurace prostředí *(Obrázek 2.1)*

Obrázek 2.1: *Konfigurace prostředí OMNeT++*

6. Simulační prostředí po úspěšné konfiguraci spustíme souborem „omnetpp.exe“ (např. C:/omnetpp-4.4.1/ide/omnetpp.exe). Zobrazí se nám uvítací okno, kde můžeme zvolit, kam chceme pokračovat (Obrázek 2.2).

Obrázek 2.2: *První spuštění OMNeT++*

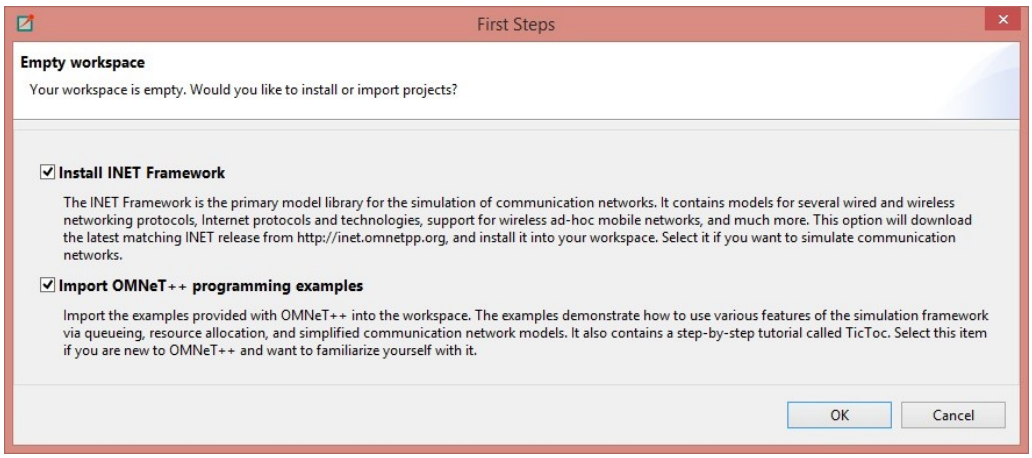
2.2 Přidání frameworku INET do prostředí OMNeT++

2.2.1 Úvod

Framework INET je rozšíření pro simulační nástroj OMNeT++. Obsahuje mnoho modulů, jako například směrovače nebo síťové protokoly, sloužící pro simulaci počítačových sítí v simulátoru. S využitím INETu tak odpadá nutnost implementovat vlastnoručně jednotlivé prvky/části sítě. Jeho součástí jsou i směrovací protokoly OSPF, BGP a v nové verzi (INET 2.3.0) je podporován i protokol RIP.

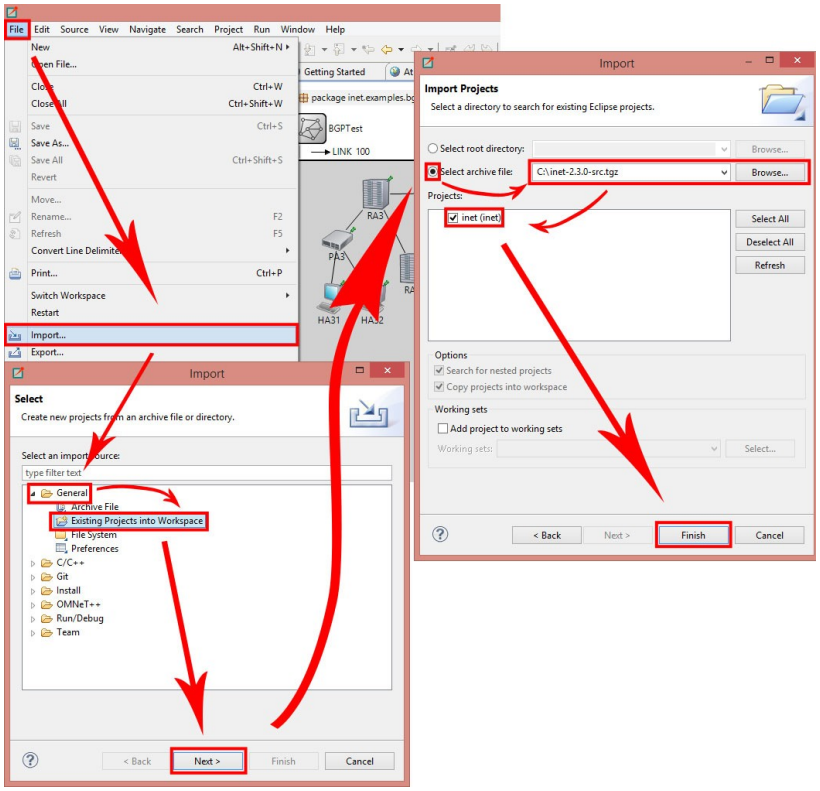
2.2.2 Postup instalace INETu

Jedním z možných postupů, jak importovat INET do simulátoru, je jeho automatická instalace. Simulátor tuto možnost nabídne po jeho prvním spuštění (Obrázek 2.3). Problémem však je fakt, že nemusí být nainstalována nejaktuálnější verze INETu a tudíž nemusí být dostupné všechny moduly – například protokol RIP.



Obrázek 2.3: *Automatická instalace modulu INET*

Druhým způsobem instalace INETu je ruční import do OMNeTu. Toto se provede následujícím způsobem (Obrázek 2.4):



Obrázek 2.4: *Importování modulu INET*

1. Stáhneme nejnovější verzi modulu INET ze stránek projektu [10]
2. V prostředí OMNeT++ importujeme modul – `File -> Import -> General -> Existing Projects into Worspace ->Select archive file ->` Zadáme cestu k souboru modulu INET -> Tlačítkem „Finish“ potvrdíme import
3. Sestavíme modul INET – v OMNeTu klikneme pravým tlačítkem myši na projekt INET a zvolíme „Build project“
4. Proběhne překlad celého frameworku

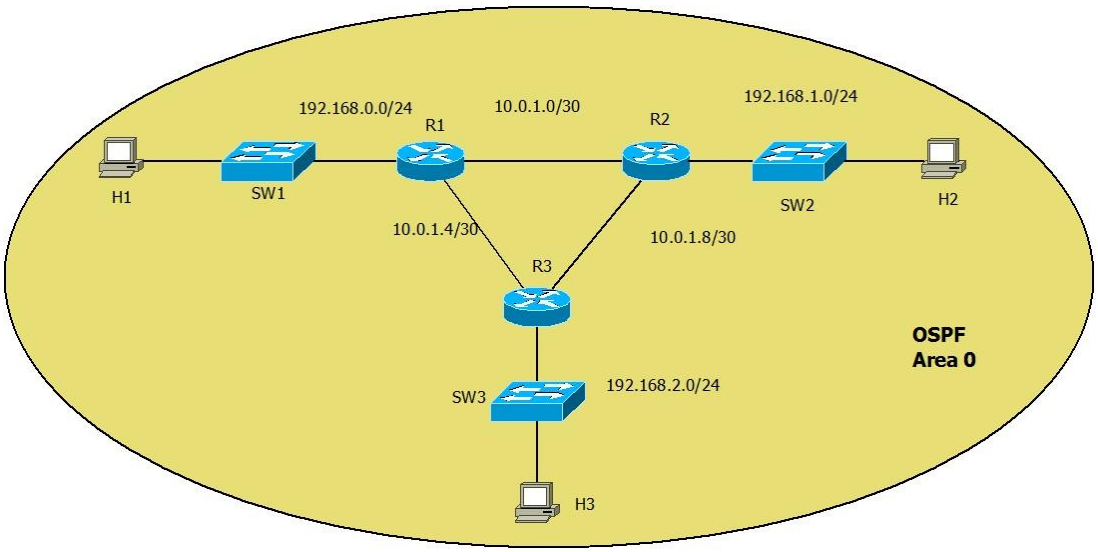
V době psaní této práce je aktuální pro Windows verze INET 2.3.0, která bohužel obsahuje chyby, takže překlad (bod 3.) se pravděpodobně nezdaří. Je nutné stáhnout dodatečné záplaty, aktualizovat některé soubory a poté se znovu pokusit o překlad. Postup je následující:

1. Ze stránek `https://github.com/inet-framework/inet/tree/v2.3.0` stáhneme zip soubor kliknutím na tlačítko „Download Zip“
2. Extrahujeme zazipovaný soubor
3. V nově vzniklé složce najdeme složky `src` a `tests`. Tyto dvě složky zkopírujeme (a nahradíme stejnojmenné soubory) pomocí Windows Průzkumníku do modulu INET importovaného do prostředí OMNeT++ - např. `C:/omnetpp-4.4.1/samples/inet`
4. Jakmile jsme nahradily staré soubory novými, můžeme se pokusit v prostředí OMNeT++ znovu sestavit modul INET

3 Aplikace směrovacích protokolů v prostředí OMNeT++

Tato kapitola obsahuje vzorové příklady aplikace směrovacích protokolů v simulátoru OMNeT++. Každý příklad je popsán krok za krokem, které je potřeba provést pro simulaci sítě.

3.1 Aplikace směrovacího protokolu OSPF



Obrázek 3.1: *Síť OSPF*

3.1.1 Zadání

- Vytvořte v simulačním prostředí OMNeT++ počítačovou síť podle Obrázku 3.1
- Nakonfigurujte síť pomoci modulu IPv4NetworkConfigurator
- Jako směrovací protokol použijte protokol OSPF
- Vytvořte v síti komunikaci přes UDP protokol
- Vytvořte události na přerušení linky a její opětovné vytvoření a sledujte chování směrovačů

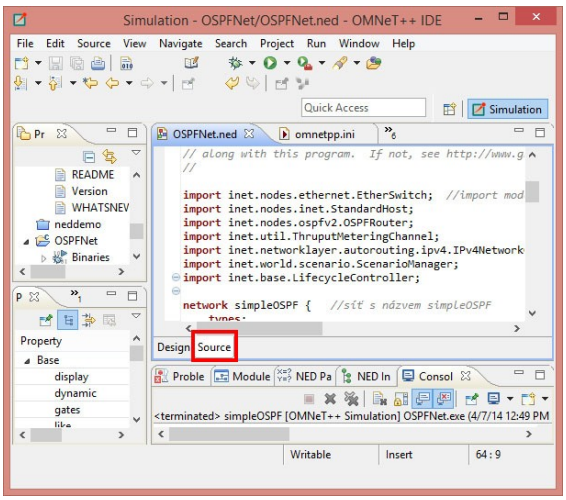
3.1.2 Založení nového projektu

Jako první založíme v OMNeTu nový projekt: `File -> New -> OMNeT++ Project`. Projekt pojmenujeme `OSPFNet`. Abychom v našem projektu mohli používat modely z frameworku INET, je nutné přidat referenci na INET: `Klikneme pravým tlačítkem myši na projekt -> Properties -> Project references -> zaškrtneme INET -> OK`.

3.1.3 Definice sítě v NED souboru

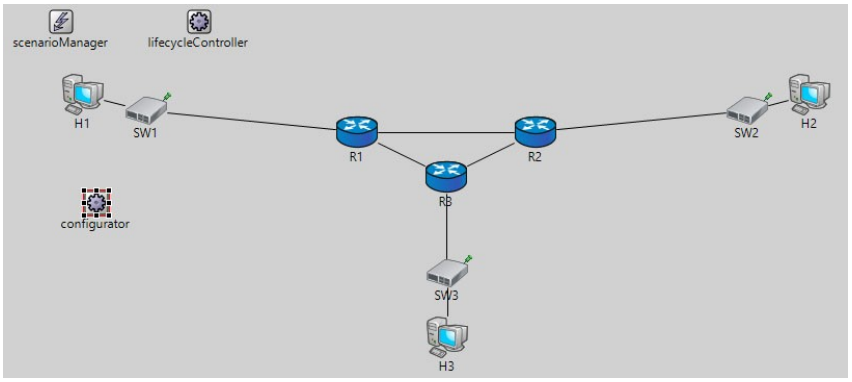
Dalším krokem je vytvoření samotné sítě. K tomuto účelu slouží soubory NED (Network DEscription), ve kterém specifikujeme detaily topologie. Vytvoříme tedy nový soubor s příponou .ned: `Pravým kliknutím myší na náš projekt -> New -> Network Description File.` Název našeho souboru zvolíme `OSPFNet.ned`.

Soubory NED můžeme editovat jak v grafickém režimu, tak můžeme upravovat samotný zdrojový kód. Přepínat mezi oběma režimy lze v levém dolním rohu editoru (Obrázek 3.2). Pro naše potřeby budeme používat textový editor a zdrojový kód.



Obrázek 3.2: *Výběr mezi grafickým a textovým editorem*

Máme tedy vytvořený NED soubor. Dříve, než začneme vytvářet topologii, musíme importovat modely, které budeme v síti potřebovat – jsou to třeba modely `StandardHost`. `OSPFRouter` nebo `IPv4NetworkConfigurator`. Poté můžeme přejít k samotné tvorbě sítě. Nejdříve si vytvoříme síť, kterou nazveme `simpleOSPF`. Pokračujeme definicí kanálu (`Channel`), který určuje vlastnosti linek mezi jednotlivými prvky sítě. Následuje vytvoření zmiňovaných síťových prvků – směrovačů, přepínačů a počítačů. Důležitou součástí je také model `configurator`, který má na starosti například nastavování IP adres nebo cest. Aby byla celá síť živější, použijeme ještě modely `LifecycleCotroller` a `ScenarioManager`, pomocí kterých můžeme naplánovat třeba vypnutí směrovače a donutit tak OSPF ke konvergenci. A jako poslední definujeme spojení (`connections`) mezi zařízeními. Grafická podoba sítě je na Obrázku 3.3:



Obrázek 3.3: *Grafická podoba souboru OSPFNet.ned*

Tomuto odpovídá tento zdrojový kód (viz. CD adresář `ospf` soubor `OSPFNet.ned`):

```
import inet.nodes.ethernet.EtherSwitch;  //import modelů
import inet.nodes.inet.StandardHost;
import inet.nodes.ospfv2.OSPFRouter;
import inet.util.ThruputMeteringChannel;
import inet.networklayer.autorouting.ipv4.IPv4NetworkConfigurator;
import inet.world.scenario.ScenarioManager;
import inet.base.LifecycleController;

network simpleOSPF {   //síť s názvem simpleOSPF
    types:
   //kanál C, který použijeme při propojování zařízení
        channel C extends ThruputMeteringChannel {
            delay = 0.1us;
            datarate = 100Mbps;
            thruputDisplayFormat = "#N";
        }

    submodules:
 R1: OSPFRouter { gates: ethg[3]; } //vytvoření OSPF směrovače
 //směrovač má 3 ethernetové rozhraní
        R2: OSPFRouter {gates: ethg[3]; }
        R3: OSPFRouter {gates: ethg[3]; }
        H1: StandardHost {gates: ethg[1]; }
        H2: StandardHost {gates: ethg[1]; }
        H3: StandardHost {gates: ethg[1]; }
        SW1: EtherSwitch {gates: ethg[2]; }
        SW2: EtherSwitch {gates: ethg[2]; }
        SW3: EtherSwitch {gates: ethg[2]; }

 scenarioManager: ScenarioManager {} //plánování událostí v simulaci
        lifecycleController: LifecycleController {} //umožňuje např. vyp/zap směrovač

        //configurator - automatické nastavení sítě
        configurator: IPv4NetworkConfigurator {
            parameters:
                config = xml("<config>"+
        "<interface among='H1 R1' address='192.168.0.x' netmask='255.255.255.0' />"+
        "<interface among='H2 R2' address='192.168.1.x' netmask='255.255.255.0' />"+
        "<interface among='H3 R3' address='192.168.2.x' netmask='255.255.255.0' />"+
        "<interface among='R1 R2' address='10.0.0.x' netmask='255.255.255.252' />"+
        "<interface among='R1 R3' address='10.0.0.x' netmask='255.255.255.252' />"+
        "<interface among='R2 R3' address='10.0.0.x' netmask='255.255.255.252' />"+
        "<route hosts='H*' destination='*' netmask='0.0.0.0' interface='eth0' />"+
                            "</config>");
                addStaticRoutes = false;
                addDefaultRoutes = false;
        }
```

```
    connections: //spojení mezi prvky; používáme i kanál C
        R1.ethg[0] <--> C <--> SW1.ethg[0]; SW1.ethg[1] <--> C <--> H1.ethg[0];
        R2.ethg[0] <--> C <--> SW2.ethg[0]; SW2.ethg[1] <--> C <--> H2.ethg[0];
        R3.ethg[0] <--> C <--> SW3.ethg[0]; SW3.ethg[1] <--> C <--> H3.ethg[0];
        R1.ethg[1] <--> C <--> R2.ethg[1]; R3.ethg[1] <--> C <--> R1.ethg[2];
        R3.ethg[2] <--> C <--> R2.ethg[2];
}
```

3.1.4 Inicializační soubor omnetpp.ini

Inicializační soubor s příponou ini slouží k nastavení parametrů simulace. Můžeme zde vytvářet například události, které se spustí v určitý čas simulace. My vytvoříme jednoduchou UDP komunikaci.

Přidáme tedy do našeho projektu ini soubor: `Pravým kliknutím myší na náš projekt -> New -> Initialization File(ini)` a pojmenujeme ho `omnetpp.ini`. Opět máme dvě možnosti, jak upravovat ini soubory – pomocí formuláře nebo úpravou zdrojového kódu. Pro naši síť nastavíme název sítě, přidáme odkaz na konfigurační xml soubor pro směrovací protokol OSPF. V další části souboru vygenerujeme nějaký síťový provoz – UDP zprávy, které se budou posílat v pravidelných intervalech. A nakonec načteme nastavení scenarioManageru zase z xml souboru (viz. CD adresář `ospf` soubor `omnetpp.ini`):

```
[General]
network = simpleOSPF
**.ospf.ospfConfig = xmldoc("Config.xml")

**.numUdpApps = 2
**.udpApp[0].typename = "UDPBasicApp"
**.udpApp[0].destPort = 1234
**.udpApp[0].messageLength = 32 bytes
**.udpApp[0].sendInterval = 0.1s
**.udpApp[0].startTime = 4s
**.H2.udpApp[0].destAddresses = "H1"
**.H1.udpApp[0].destAddresses = "H2"
**.udpApp[1].typename = "UDPEchoApp"
**.udpApp[1].localPort = 1234

*.scenarioManager.script = xmldoc("scenario.xml")
```

3.1.5 Konfigurační soubor protokolu OSPF

Pro nastavení směrování na všech směrovačích můžeme použít pouze jeden xml soubor. Vytvoříme ho podobným způsobem, jako jsme do projektu přidávali ini a ned soubory: `Pravým kliknutím myší na náš projekt -> New -> File`. Soubor pojmenujeme `Config.xml`. Definujeme v něm oblasti OSPF, v našem případě pouze jedinou oblast 0.0.0.0, a také například nastavíme rozhraní směrovačů, či jejich cenu (viz. CD adresář `ospf` soubor `Config.xml`):

```
<?xml version="1.0"?>
<OSPFASConfig >
  <!-- Areas -->
  <Area id="0.0.0.0">
    <AddressRange address="H1" mask="H1" status="Advertise" />
    <AddressRange address="H2" mask="H2" status="Advertise" />
    <AddressRange address="H3" mask="H3" status="Advertise" />
```

```
    <AddressRange address="R1>R2" mask="R1>R2" status="Advertise" />
    <AddressRange address="R2>R1" mask="R2>R1" status="Advertise" />
    <AddressRange address="R1>R3" mask="R1>R3" status="Advertise" />
    <AddressRange address="R2>R3" mask="R2>R3" status="Advertise" />
    <AddressRange address="R3>R2" mask="R3>R2" status="Advertise" />
    <AddressRange address="R3>R1" mask="R3>R1" status="Advertise" />
  </Area>

  <!-- Routers -->
  <Router name="R1" RFC1583Compatible="true">
    <PointToPointInterface ifName="eth1" areaID="0.0.0.0" interfaceOutputCost="1" />
    <PointToPointInterface ifName="eth2" areaID="0.0.0.0" interfaceOutputCost="1" />
    <BroadcastInterface ifName="eth0" areaID="0.0.0.0" interfaceOutputCost="2"
routerPriority="2" />
  </Router>

  <Router name="R2" RFC1583Compatible="true">
    <PointToPointInterface ifName="eth1" areaID="0.0.0.0" interfaceOutputCost="2" />
    <PointToPointInterface ifName="eth2" areaID="0.0.0.0" interfaceOutputCost="1" />
    <BroadcastInterface ifName="eth0" areaID="0.0.0.0" interfaceOutputCost="1"
routerPriority="2" />
  </Router>

  <Router name="R3" RFC1583Compatible="true">
    <PointToPointInterface ifName="eth1" areaID="0.0.0.0" interfaceOutputCost="4" />
    <PointToPointInterface ifName="eth2" areaID="0.0.0.0" interfaceOutputCost="4" />
    <BroadcastInterface ifName="eth0" areaID="0.0.0.0" interfaceOutputCost="1"
routerPriority="2" />
  </Router>
</OSPFASConfig>
```

3.1.6 **Naplánování událostí v simulaci**

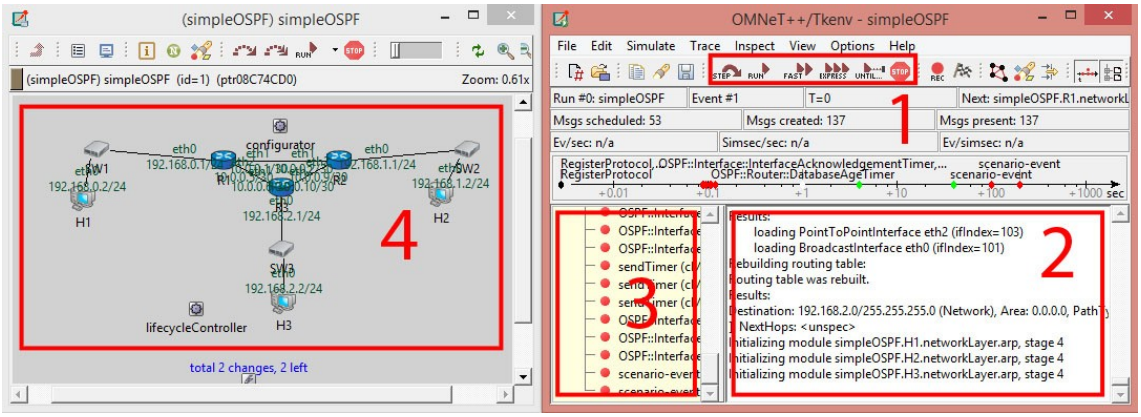
V simulaci můžeme naplánovat určité události, které se spustí v určitý čas. Jedním způsobem, jak nějakou událost naplánovat, je použití scenarioManageru. Ten jsme definovali v ned souboru, v ini souboru jsme zadali název souboru s nastavením a teď je načase konfigurační soubor vytvořit. Vytvoříme tedy další xml: `Pravým kliknutím myší na náš projekt -> New -> File.` Tentokrát ho pojmenujeme `scenario.xml` a v něm vytvoříme 2 události - v čase t=100 zrušíme linku mezi R1 a R2 a v čase t=200 linku mezi R1 a R2 obnovíme (viz. CD adresář `ospf` soubor `scenario.xml`):

```
<scenario>
    <at t="100">
        <disconnect src-module="R1" src-gate="ethg$o[1]" />
        <disconnect src-module="R2" src-gate="ethg$o[1]" />
    </at>
    <at t="200">
        <connect src-module="R1" src-gate="ethg[1]"
                 dest-module="R2" dest-gate="ethg[1]"
                 channel-type="inet.util.ThruputMeteringChannel">
            <param name="delay" value="0.1us" />
            <param name="datarate" value="100Mbps" />
            <param name="thruputDisplayFormat" value='"#N"' />
        </connect>
    </at>
</scenario>
```

`Scenario.xml` byl poslední soubor, který jsme potřebovali. Teď máme projekt připravený k simulaci.

3.1.7 Simulace

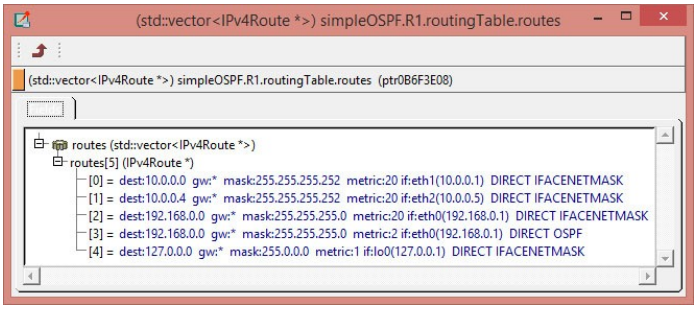
Projekt spustíme zeleným tlačítkem `Run` ve vrchní liště OMNeTu. Jestliže projekt spouštíme poprvé, nebo došlo od posledního spuštění k nějakým změnám, proběhne nejdříve překlad a až poté se spustí simulační rozhraní.



Obrázek 3.4: *Vzhled simulačního prostředí*

Po startu simulace se objeví dvě okna (Obrázek 3.4) – v pravé části můžeme vidět ovládání simulace – krokování, spuštění, atp. (1), dále zprávy informující o právě proběhnutých operacích/událostech v simulaci (2) a v levé části pravého okna vidíme nadcházející události (3) – např. naplánované události směrovacího protokolu, nebo události, které jsme nakonfigurovali dříve v scenarioManageru. V levém okně pak vidíme grafické zobrazení sítě a aktuální dění (4).

Na jednotlivé prvky v síti můžeme poklepat levým tlačítkem myši a zobrazit tak jejich parametry a vlastnosti. Jestliže třeba chceme zobrazit směrovací tabulku směrovače R1, poklepeme myší na směrovač, zobrazí se nám jeho vnitřní struktura, kde tentokrát poklepeme na `routing table` a následně na vektor `Routes`. Takto jsme se proklikali do směrovací tabulky (Obrázek 3.5), kde vidíme (zatím) pouze záznamy o přímo připojených sítích. Jakmile spustíme simulaci tlačítkem  , po nějaké době dojde ke konvergenci sítě a směrovací tabulky, stejně tak jako další parametry sítě, se změní.



Obrázek 3.5: *Směrovací tabulka R1*

Nyní spustíme simulaci a po nějaké chvíli ji zase zastavíme, abychom se podívali, co se v síti událo.

```
** Event #26  T=0.089228422814  simpleOSPF.R2.networkLayer.ip (IPv4, id=46), on `OSPF_HelloPacket' (OSPFHelloPacket, id=137)
Sending datagram `OSPF_HelloPacket' with dest=224.0.0.5
multicast packet routed by socket option via output interface eth0
destination address is multicast, sending packet to MAC address 01-00-5E-00-00-05
Sending out packet to interface eth0
```

Obrázek 3.6: *Událost č.26*

Na obrázku 3.6 můžeme vidět událost, která byla v pořadí 26. a nastala v čase T. Šlo o vyslání Hello paketu ze směrovače R2 rozhraním eth0. Cílová IP adresa byla 224.0.0.5.

Další událost, kterou si rozebereme, je naše plánované odstavení linky. Najdeme si tedy událost v okně s událostmi – `scenario-event, at T=100`, klikneme pravým tlačítkem a zvolíme možnost „Express run until message“. Tím se simulace znovu spustí a zastaví se těsně před vykonáním zvolené události. Podíváme se, co obsahuje směrovací tabulka směrovače R1 (Obrázek 3.7).

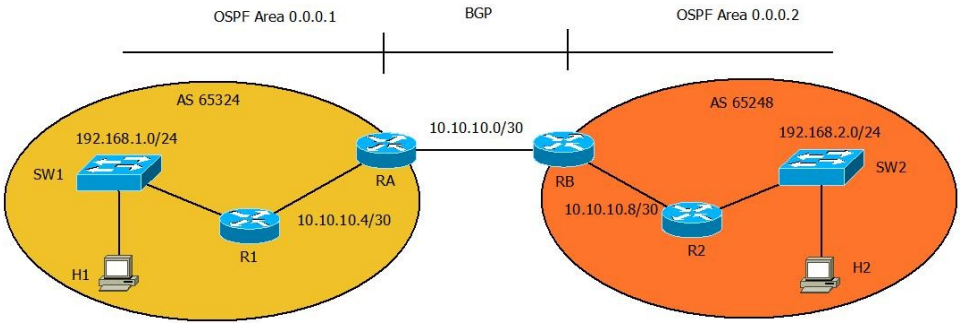
Obrázek 3.7: *Směrovací tabulka směrovače R1 před odpojením linky*

Jestliže se v simulaci posuneme o jeden krok, dojde k odpojení linky a tím přestanou být některé cesty platné. Směrovače tento fakt zaregistrují a začnou aktualizovat své tabulky. Posuneme tedy simulaci o nějaký čas vpřed a znovu se podíváme na směrovací tabulku (Obrázek 3.8)

Obrázek 3.8: *Směrovací tabulka směrovače R1 po odpojení linky*

Když porovnáme tabulky před a po odpojení linky, zjistíme, že některé cesty se změnily, a směrovače a OSPF fungují, jak mají.

3.2 Aplikace směrovacího protokolu BGP



Obrázek 3.9: *BGP síť*

3.2.1 Zadání

- Vytvořte v simulátoru OMNeT++ počítačovou síť podle Obrázku 3.9
- Mezi směrovači RA a RB aplikujte směrovací protokol BGP
- V krajních sítích aplikujte protokol OSPF
- Vygenerujte UDP provoz

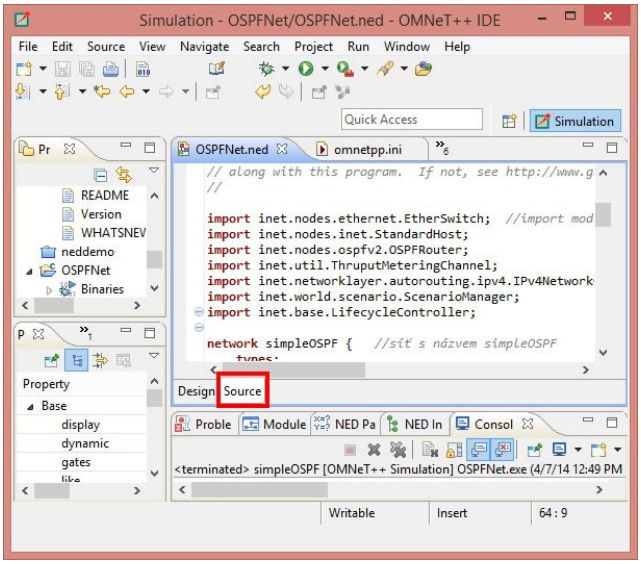
3.2.2 Založení nového projektu

Jako první založíme v OMNeTu nový projekt: `File -> New -> OMNeT++ Project`. Projekt pojmenujeme `BGPNet`. Abychom v našem projektu mohli používat modely z frameworku INET, je nutné přidat referenci na INET: `Klikneme pravým tlačítkem myši na projekt -> Properties -> Project references -> zaškrtneme INET -> OK.`

3.2.3 Definice sítě v NED souboru

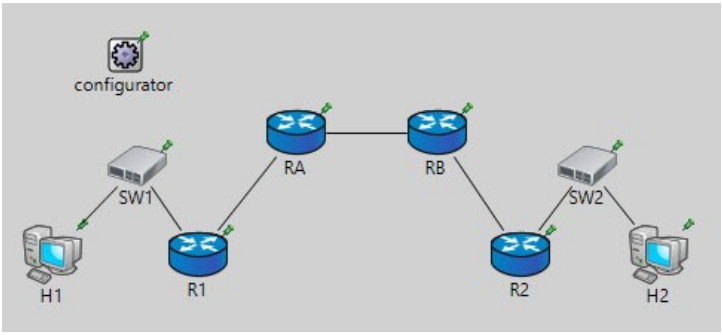
Dalším krokem je vytvoření samotné sítě. K tomuto účelu slouží soubory NED (Network DEscription), ve kterém specifikujeme detaily topologie. Vytvoříme tedy nový soubor s příponou .ned: `Pravým kliknutím myší na náš projekt -> New -> Network Description File.` Název našeho souboru zvolíme `BGPNet.ned`.

Soubory NED můžeme editovat jak v grafickém režimu, tak můžeme upravovat samotný zdrojový kód. Přepínat mezi oběma režimy lze v levém dolním rohu editoru (Obrázek 3.10). Pro naše potřeby budeme používat textový editor a zdrojový kód.



Obrázek 3.10: *Výběr mezi grafickým a textovým editorem*

Máme tedy vytvořený NED soubor. Dříve, než začneme vytvářet topologii, musíme importovat modely, které budeme v síti potřebovat – jsou to třeba modely `StandardHost`. `OSPFRouter` nebo `BGPRouter`. Poté můžeme přejít k samotné tvorbě sítě. Nejdříve si vytvoříme síť, kterou nazveme `simpleBGP`. Pokračujeme definicí kanálu (`Channel`), který určuje vlastnosti linek mezi jednotlivými prvky sítě. Následuje vytvoření zmiňovaných síťových prvků – směrovačů, přepínačů a počítačů. Důležitou součástí je také model `configurator`, který má na starosti například nastavování IP adres nebo cest. A jako poslední definujeme spojení (`connections`) mezi zařízeními. Naše síť může vypadat podobně jako na obrázku 3.11.



Obrázek 3.11: *Grafická podoba souboru BGPNet.ned*

Tomuto odpovídá tento zdrojový kód (viz. CD adresář `bgp` soubor `BGPNet.ned`):

```
import inet.nodes.bgp.BGPRouter;
import inet.nodes.ethernet.EtherSwitch;
import inet.util.ThruputMeteringChannel;
import inet.networklayer.autorouting.ipv4.IPv4NetworkConfigurator;
import inet.nodes.inet.StandardHost;
import inet.nodes.ospfv2.OSPFRouter;
```

```
network simpleBGP
{
    types:
        channel C extends ThruputMeteringChannel {
            delay = 0.1us;
            datarate = 100Mbps;
            thruputDisplayFormat = "#N";
        }

    submodules:
        RA: BGPRouter {gates: pppg[1]; ethg[1]; }
        RB: BGPRouter {gates: pppg[1]; ethg[1]; }
        R1: OSPFRouter {gates: ethg[2]; }
        R2: OSPFRouter {gates: ethg[2]; }
        SW1: EtherSwitch {gates: ethg[2];}
        SW2: EtherSwitch {gates: ethg[2];}
        H1: StandardHost {gates: ethg[1];}
        H2: StandardHost {gates: ethg[1];}

        configurator: IPv4NetworkConfigurator {
            @display("p=94,54");
            config = xmldoc("conf.xml");
            addStaticRoutes = false;
            addDefaultRoutes = false;
            addSubnetRoutes = false;
        }
    connections:
        RA.pppg[0] <--> C <--> RB.pppg[0]; RA.ethg[0] <--> C <--> R1.ethg[0];
        RB.ethg[0] <--> C <--> R2.ethg[0]; R1.ethg[1] <--> C <--> SW1.ethg[0];
        R2.ethg[1] <--> C <--> SW2.ethg[0]; SW1.ethg[1] <--> C <--> H1.ethg[0];
        SW2.ethg[1] <--> C <--> H2.ethg[0];
}
```

3.2.4 Inicializační soubor omnetpp.ini

Inicializační soubor s příponou ini slouží k nastavení parametrů simulace. Můžeme zde vytvářet například události, které se spustí v určitý čas simulace. My vytvoříme jednoduchou UDP komunikaci.

Přidáme tedy do našeho projektu ini soubor: `Pravým kliknutím myší na náš projekt -> New -> Initialization File(ini)` a pojmenujeme ho omnetpp.ini. Opět máme dvě možnosti, jak upravovat ini soubory – pomocí formuláře nebo úpravou zdrojového kódu. Pro naši síť nastavíme název sítě, přidáme odkaz na konfigurační xml soubor pro směrovací protokoly BGP a OSPF. V další části souboru vygenerujeme nějaký síťový provoz – UDP zprávy, které se budou posílat v pravidelných intervalech (viz. CD adresář `bgp` soubor `omnetpp.ini`):

```
[General]
network = simpleBGP

**.bgp.dataTransferMode = "object"

# OSPF configuration
**.ospfConfig = xmldoc("OSPFconf.xml")

# bgp settings
**.bgpConfig = xmldoc("BGPconf.xml")

**.numUdpApps = 2
**.udpApp[0].typename = "UDPBasicApp"
**.udpApp[0].destPort = 1234
**.udpApp[0].messageLength = 32 bytes
**.udpApp[0].sendInterval = 0.1s
```

```
**.udpApp[0].startTime = 4s
**.H2.udpApp[0].destAddresses = "H1"
**.H1.udpApp[0].destAddresses = "H2"
**.udpApp[1].typename = "UDPEchoApp"
**.udpApp[1].localPort = 1234
```

3.2.5 **Konfigurační soubor protokolu OSPF**

Pro nastavení směrování na všech směrovačích můžeme použít pouze jeden xml soubor. Vytvoříme ho podobným způsobem, jako jsme do projektu přidávali ini a ned soubory: `Pravým kliknutím myší na náš projekt -> New -> File`. Soubor pojmenujeme OSPFconf.xml. Definujeme v něm oblasti OSPF, 0.0.0.1 a 0.0.0.2, a také například nastavíme rozhraní směrovačů (viz. CD adresář `bgp` soubor `OSPFconf.xml`):

```
<?xml version="1.0"?>
<OSPFASConfig>
    <Area id="0.0.0.1">
        <AddressRange address="10.10.10.4" mask="255.255.255.252" status="Advertise" />
    </Area>
    <Area id="0.0.0.2">
        <AddressRange address="10.10.10.8" mask="255.255.255.252" status="Advertise" />
    </Area>
    <Router name="R1" RFC1583Compatible="true">
       <PointToPointInterface ifName="eth0" areaID="0.0.0.1" interfaceOutputCost="10" />
        <ExternalInterface ifName="eth1" advertisedExternalNetworkAddress="192.168.1.0"
advertisedExternalNetworkMask="255.255.255.0"
externalInterfaceOutputCost="10" externalInterfaceOutputType="Type2"
forwardingAddress="0.0.0.0" externalRouteTag="0x00" />
    </Router>
    <Router name="RA" RFC1583Compatible="true">
       <PointToPointInterface ifName="eth0" areaID="0.0.0.1" interfaceOutputCost="10" />
    </Router>
    <Router name="R2" RFC1583Compatible="true">
       <PointToPointInterface ifName="eth0" areaID="0.0.0.2" interfaceOutputCost="10" />
        <ExternalInterface ifName="eth1" advertisedExternalNetworkAddress="192.168.2.0"
advertisedExternalNetworkMask="255.255.255.0"
externalInterfaceOutputCost="10" externalInterfaceOutputType="Type2"
forwardingAddress="0.0.0.0" externalRouteTag="0x00" />
    </Router>
    <Router name="RB" RFC1583Compatible="true">
       <PointToPointInterface ifName="eth0" areaID="0.0.0.2" interfaceOutputCost="10" />
    </Router>
</OSPFASConfig>
```

3.2.6 **Konfigurační soubor protokolu BGP**

Teď nastavíme protokol BGP. Vytvoříme další konfigurační xml soubor, tentokrát soubor pojmenujeme BGPconf.xml. Nastavíme Autonomní systémy a taky parametry směrovacího protokolu – časovače (viz. CD adresář `bgp` soubor `BGPconf.xml`):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BGPConfig>
    <TimerParams>
        <connectRetryTime> 120 </connectRetryTime>
        <holdTime> 180 </holdTime>
        <keepAliveTime> 60 </keepAliveTime>
        <startDelay> 15 </startDelay>
    </TimerParams>
    <AS id="65324">
        <Router interAddr="10.10.10.1"/> <!--router RA-->
    </AS>
    <AS id="65248">
```

```
        <Router interAddr="10.10.10.2"/> <!--router RB-->
    </AS>
   <Session id="1">
        <Router exterAddr="10.10.10.2"/>
        <Router exterAddr="10.10.10.1"/>
    </Session>
</BGPConfig>
```

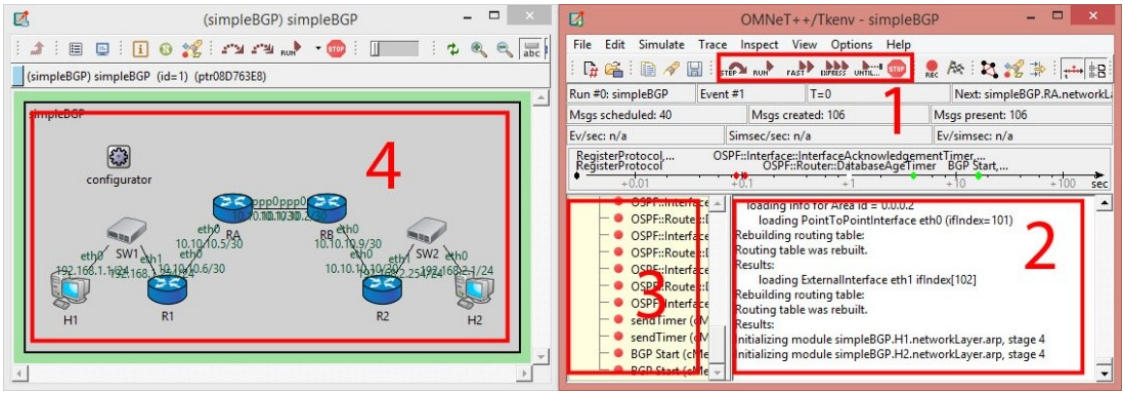
3.2.7 Konfigurační soubor pro IPv4NetworkConfigurator

Pro IPv4NetworkConfigurator použijeme externí soubor xml. V něm nastavíme jednotlivé rozhraní síťových prvků (viz. CD adresář `bgp` soubor `conf.xml`):

```
<?xml version="1.0"?>
       <config>
         <interface hosts='RA' names='ppp0' address='10.10.10.1'
        netmask='255.255.255.252' groups='224.0.0.1 224.0.0.2 224.0.0.5' metric='1'/>
         <interface hosts='RA' names='eth0' address='10.10.10.5'
        netmask='255.255.255.252' groups='224.0.0.1 224.0.0.2 224.0.0.5' metric='1'/>
         <interface hosts='RB' names='ppp0' address='10.10.10.2'
        netmask='255.255.255.252' groups='224.0.0.1 224.0.0.2 224.0.0.5' metric='1'/>
         <interface hosts='RB' names='eth0' address='10.10.10.9'
        netmask='255.255.255.252' groups='224.0.0.1 224.0.0.2 224.0.0.5' metric='1'/>
         <interface hosts='R1' names='eth0' address='10.10.10.6'
        netmask='255.255.255.252' groups='224.0.0.1 224.0.0.2 224.0.0.5' metric='1'/>
         <interface hosts='R1' names='eth1' address='192.168.1.254'
        netmask='255.255.255.0' metric='1'/>
         <interface hosts='R2' names='eth0' address='10.10.10.10'
        netmask='255.255.255.252' groups='224.0.0.1 224.0.0.2 224.0.0.5' metric='1'/>
         <interface hosts='R2' names='eth1' address='192.168.2.254'
        netmask='255.255.255.0' metric='1'/>
         <interface hosts='H1' names='eth0' address='192.168.1.1'
       netmask='255.255.255.0' mtu='1500' metric='1'/>
         <interface hosts='H2' names='eth0' address='192.168.2.1'
        netmask='255.255.255.0' mtu='1500' metric='1'/>
         <route hosts='H*' destination='*' netmask='*' interface='eth0'/>
       </config>
```

3.2.8 Simulace

Projekt spustíme zeleným tlačítkem `Run` ve vrchní liště OMNeTu. Jestliže projekt spouštíme poprvé, nebo došlo od posledního spuštění k nějakým změnám, proběhne nejdříve překlad a až poté se spustí simulační rozhraní.



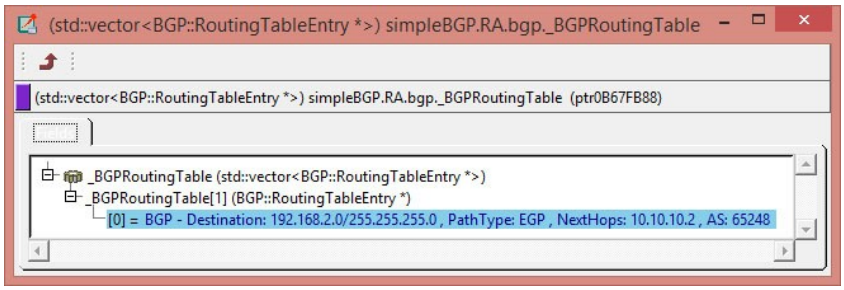
Obrázek 3.12: *Vzhled simulačního prostředí*

Po startu simulace se objeví dvě okna (Obrázek 3.12) – v pravé části můžeme vidět ovládání simulace – krokování, spuštění, atp. (1), dále zprávy informující o právě proběhnutých

operacích/událostech v simulaci (2) a v levé části pravého okna vidíme nadcházející události (3) – např. naplánované události směrovacího protokolu, nebo UDP komunikaci. V levém okně pak vidíme grafické zobrazení sítě a aktuální dění (4).

Na jednotlivé prvky v síti můžeme poklepat levým tlačítkem myši a zobrazit tak jejich parametry a vlastnosti. Jestliže třeba chceme zobrazit směrovací tabulku BGP směrovače RA, poklepeme myší na směrovač, zobrazí se nám jeho vnitřní struktura, kde tentokrát poklepeme na `bgp` a následně na vektor `_BGPRoutingTable`. Takto jsme se proklikali do směrovací tabulky, kde zatím nevidíme žádný záznam. Jakmile spustíme simulaci tlačítkem 

, po určitém čase si směrovače vymění BGP zprávy a naplní své směrovací tabulky

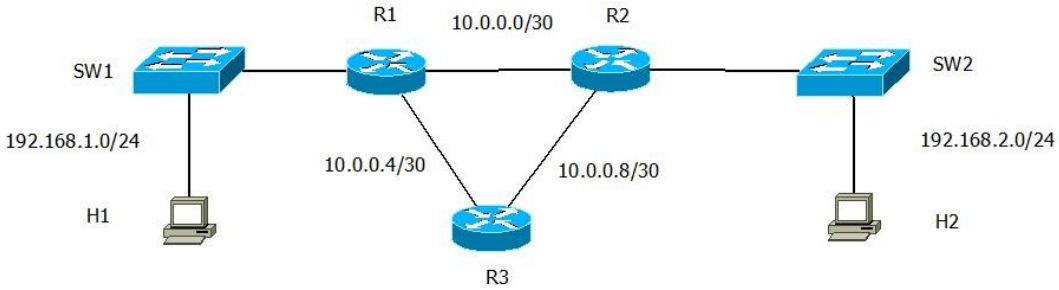


Obrázek 3.13: *Směrovací tabulka BGP směrovače RA*

Obrázek 3.13 ukazuje obsah směrovací tabulky protokolu BGP směrovače RA. Vidíme pouze jednu cestu do sítě 192.168.2.0/24 přes next-hop 10.10.10.2. Síť je z autonomního systému 65248.

Tímto jsme nasimulovali základní síť s BGP a OSPF směrováním. Projekt bychom mohli rozšířit o další směrovače, přidat další podsítě, nebo můžeme přidat nějaké záložní linky a poté některou trasu přerušit a sledovat konvergenci sítě.

3.3 Aplikace směrovacího protokolu RIP



Obrázek 3.14: *Síť pro simulaci RIP protokolu*

3.3.1 Zadání

- Vytvořte topologii sítě podle Obrázku 3.14 v simulačním prostředí OMNeT++
- V síti použijte směrovací protokol RIP

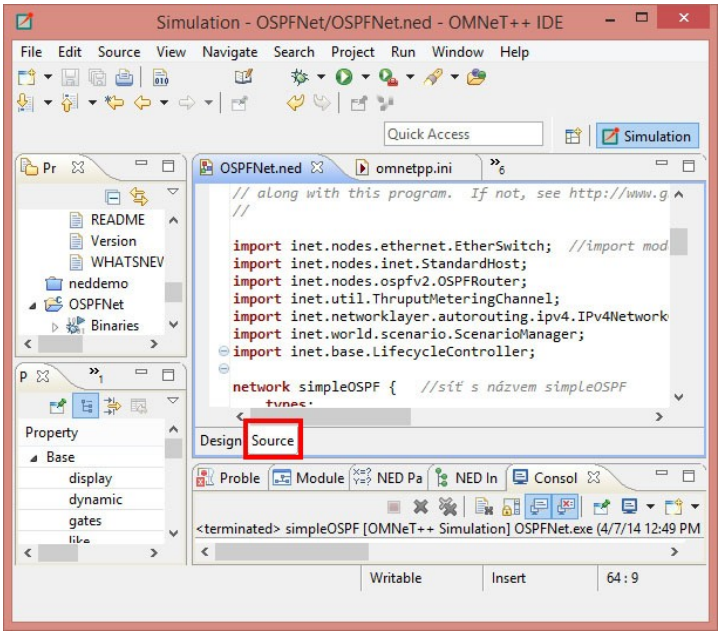
- Vygenerujte v síti UDP komunikaci mezi H1 a H2
- Vytvořte událost, která zruší spojení mezi směrovači R1 a R2, a sledujte změny ve směrovací tabulce

3.3.2 Založení nového projektu

Jako první založíme v OMNeTu nový projekt: `File -> New -> OMNeT++ Project`. Projekt pojmenujeme RIPNet. Abychom v našem projektu mohli používat modely z frameworku INET, je nutné přidat referenci na INET: `Klikneme pravým tlačítkem myši na projekt -> Properties -> Project references -> zaškrtneme INET -> OK.`

3.3.3 Definice sítě v NED souboru

Dalším krokem je vytvoření samotné sítě. K tomuto účelu slouží soubory NED (Network DEscription), ve kterém specifikujeme detaily topologie. Vytvoříme tedy nový soubor s příponou .ned: `Pravým kliknutím myší na náš projekt -> New -> Network Description File.` Název našeho souboru zvolíme `RIPNet.ned`.

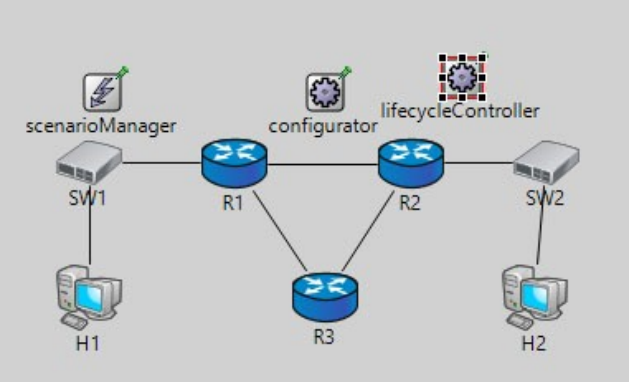


Obrázek 3.15: *Výběr mezi grafickým a textovým editorem*

Soubory NED můžeme editovat jak v grafickém režimu, tak můžeme upravovat samotný zdrojový kód. Přepínat mezi oběma režimy lze v levém dolním rohu editoru (Obrázek 3.15). Pro naše potřeby budeme používat textový editor a zdrojový kód.

Máme tedy vytvořený NED soubor. Dříve, než začneme vytvářet topologii, musíme importovat modely, které budeme v síti potřebovat – jsou to třeba modely `StandardHost`. `RIPRouter` nebo `IPv4NetworkConfigurator`. Poté můžeme přejít k samotné tvorbě

sítě. Nejdříve si vytvoříme síť, kterou nazveme `simpleRIP`. Pokračujeme definicí kanálu (`Channel`), který určuje vlastnosti linek mezi jednotlivými prvky sítě. Následuje vytvoření zmiňovaných síťových prvků – směrovačů, přepínačů a počítačů. Důležitou součástí je také model `configurator`, který má na starosti například nastavování IP adres nebo cest. Aby byla celá síť živější, použijeme ještě modely `LifecycleCotroller` a `ScenarioManager`, pomocí kterých můžeme naplánovat třeba vypnutí směrovače a donutit tak RIP ke konvergenci. A jako poslední definujeme spojení (`connections`) mezi zařízeními. Grafická podoba sítě je na Obrázku 3.16



Obrázek 3.16: *Grafická podoba souboru RIPNet.ned*

Tomuto odpovídá tento zdrojový kód (viz. CD adresář `rip` soubor `RIPNet.ned`):

```
import inet.nodes.ethernet.EtherSwitch;  //import modelů
import inet.networklayer.autorouting.ipv4.IPv4NetworkConfigurator;
import inet.nodes.inet.StandardHost;
import inet.nodes.rip.RIPRouter;
import inet.util.ThruputMeteringChannel;
import inet.world.scenario.ScenarioManager;
import inet.base.LifecycleController;

network simpleRIP{
    types:
        channel C extends ThruputMeteringChannel{
            delay = 0.1us;
            datarate = 100Mbps;
            thruputDisplayFormat = "#N";
        }

    submodules:
        R1: RIPRouter {gates: ethg[3];}
        R2: RIPRouter {gates: ethg[3];}
        R3: RIPRouter {gates: ethg[2];}
        SW1: EtherSwitch {gates:ethg[2];}
        SW2: EtherSwitch {gates:ethg[2];}
        H1: StandardHost {gates:ethg[1];}
        H2: StandardHost {gates:ethg[1];}
        scenarioManager: ScenarioManager { }
        lifecycleController: LifecycleController { }

    configurator: IPv4NetworkConfigurator {
            config = xmldoc("conf.xml");
            addStaticRoutes = false;
            addDefaultRoutes = false;
            addSubnetRoutes = false;
            @display("p=221,69");
        }
```

```
    connections:
        H1.ethg[0] <--> C <--> SW1.ethg[0];
        SW1.ethg[1] <--> C <--> R1.ethg[0];
        R1.ethg[1] <--> C <--> R2.ethg[0];
        R2.ethg[1] <--> C <--> SW2.ethg[0];
        SW2.ethg[1] <--> C <--> H2.ethg[0];
        R1.ethg[2] <--> C <--> R3.ethg[0];
        R2.ethg[2] <--> C <--> R3.ethg[1];
}
```

3.3.4 Inicializační soubor omnetpp.ini

Inicializační soubor s příponou ini slouží k nastavení parametrů simulace. Můžeme zde vytvářet například události, které se spustí v určitý čas simulace. My vytvoříme jednoduchou UDP komunikaci.

Přidáme tedy do našeho projektu ini soubor: `Pravým kliknutím myší na náš projekt -> New -> Initialization File(ini)` a pojmenujeme ho omnetpp.ini. Opět máme dvě možnosti, jak upravovat ini soubory – pomocí formuláře nebo úpravou zdrojového kódu. Pro naši síť nastavíme název sítě, přidáme odkaz na konfigurační xml soubor pro směrovací protokol RIP. V další části souboru vygenerujeme nějaký síťový provoz – UDP zprávy, které se budou posílat v pravidelných intervalech. A nakonec načteme nastavení scenarioManageru zase z xml souboru (viz. CD adresář `rip` soubor `omnetpp.ini`):

```
[General]
network = simpleRIP
**.ripConfig = xmldoc("RIPConf.xml")

**.numUdpApps = 2
**.udpApp[0].typename = "UDPBasicApp"
**.udpApp[0].destPort = 1234
**.udpApp[0].messageLength = 32 bytes
**.udpApp[0].sendInterval = 0.1s
**.udpApp[0].startTime = 4s
**.H2.udpApp[0].destAddresses = "H1"
**.H1.udpApp[0].destAddresses = "H2"
**.udpApp[1].typename = "UDPEchoApp"
**.udpApp[1].localPort = 1234

*.scenarioManager.script = xmldoc("scenario.xml")
```

3.3.5 Konfigurační soubor protokolu RIP

Pro nastavení směrování na všech směrovačích můžeme použít pouze jeden xml soubor. Vytvoříme ho podobným způsobem, jako jsme do projektu přidávali ini a ned soubory: `Pravým kliknutím myší na náš projekt -> New -> File`. Soubor pojmenujeme `RIPConf.xml`. V konfiguračním souboru nastavíme pouze metriku jednotlivých linek (viz. CD adresář `rip` soubor `RIPConf.xml`):

```
<?xml version="1.0"?>
<RIPConfig>
  <interface among="R1 R3" metric="3"/>
  <interface among="R? R?" metric="1"/>
</RIPConfig>
```

3.3.6 Konfigurační soubor pro IPv4NetworkConfigurator

I pro IPv4NetworkConfigurator použijeme externí soubor xml. Vytvoříme tedy soubor `conf.xml` a v něm nastavíme jednotlivé rozhraní síťových prvků (viz. CD adresář `rip` soubor `conf.xml`):

```
<config>
  <interface hosts='R1' names='eth0' address='10.0.0.1' netmask='255.255.255.252'/>
  <interface hosts='R1' names='eth1' address='10.0.0.5' netmask='255.255.255.252'/>
  <interface hosts='R1' names='eth2' address='192.168.1.1' netmask='255.255.255.0'/>
  <interface hosts='R2' names='eth0' address='10.0.0.2' netmask='255.255.255.252'/>
  <interface hosts='R2' names='eth1' address='10.0.0.9' netmask='255.255.255.252'/>
  <interface hosts='R2' names='eth2' address='192.168.2.1' netmask='255.255.255.0'/>
  <interface hosts='R3' names='eth0' address='10.0.0.6' netmask='255.255.255.252'/>
  <interface hosts='R3' names='eth1' address='10.0.0.10' netmask='255.255.255.252'/>
  <interface hosts='H1' names='eth0' address='192.168.1.2' netmask='255.255.255.0'
mtu='1500' metric='1'/>
  <interface hosts='H2' names='eth0' address='192.168.2.2' netmask='255.255.255.0'
mtu='1500' metric='1'/>
  <route hosts='H1' destination='*' gateway='R1'/>
  <route hosts='H2' destination='*' gateway='R2'/>
</config>
```

3.3.7 Naplánování událostí v simulaci

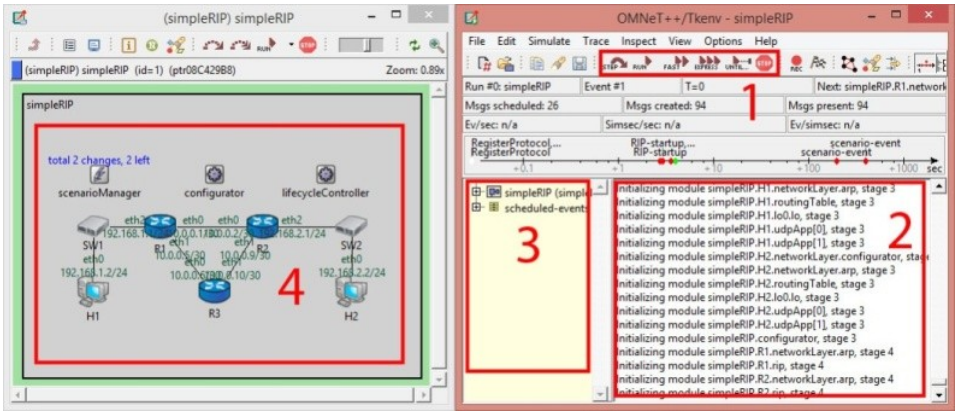
V simulaci můžeme naplánovat určité události, které se spustí v určitý čas. Jedním způsobem, jak nějakou událost naplánovat, je použití `scenarioManageru`. Ten jsme definovali v ned souboru, v ini souboru jsme zadali název souboru s nastavením a teď je načase konfigurační soubor vytvořit. Vytvoříme tedy další xml: `Pravým kliknutím myší na náš projekt -> New -> File.` Tentokrát ho pojmenujeme `scenario.xml` a v něm vytvoříme 2 události - v čase t=200 zrušíme linku mezi R1 a R2 a v čase t=400 linku mezi R1 a R2 obnovíme (viz. CD adresář `rip` soubor `scenario.xml`):

```
<scenario>
    <at t="100">
        <disconnect src-module="R1" src-gate="ethg$o[1]" />
        <disconnect src-module="R2" src-gate="ethg$o[1]" />
    </at>
    <at t="200">
        <connect src-module="R1" src-gate="ethg[1]"
                 dest-module="R2" dest-gate="ethg[1]"
                 channel-type="inet.util.ThruputMeteringChannel">
            <param name="delay" value="0.1us" />
            <param name="datarate" value="100Mbps" />
            <param name="thruputDisplayFormat" value='"#N"' />
        </connect>
    </at>
</scenario>
```

`Scenario.xml` byl poslední soubor, který jsme potřebovali. Teď máme projekt připravený k simulaci.

3.3.8 Simulace

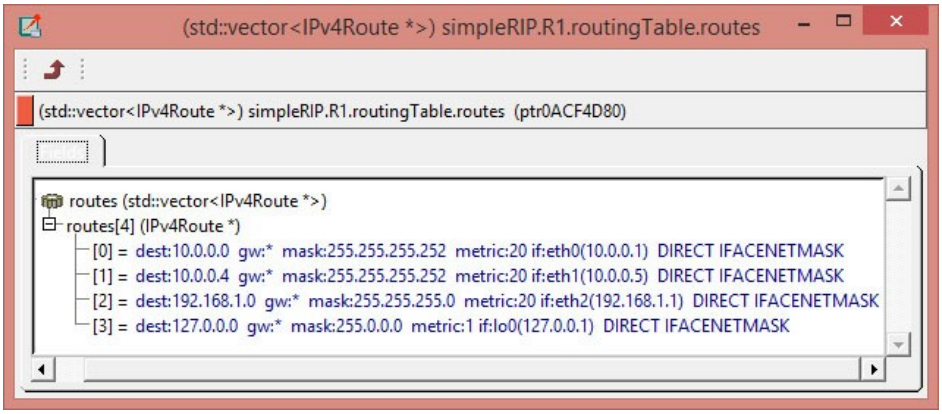
Projekt spustíme zeleným tlačítkem `Run` ve vrchní liště OMNeTu. Jestliže projekt spouštíme poprvé, nebo došlo od posledního spuštění k nějakým změnám, proběhne nejdříve překlad a až poté se spustí simulační rozhraní.



Obrázek 3.17: *Vzhled simulačního prostředí*

Po startu simulace se objeví dvě okna (Obrázek 3.17) – v pravé části můžeme vidět ovládání simulace – krokování, spuštění, atp. (1), dále zprávy informující o právě proběhnutých operacích/událostech v simulaci (2) a v levé části pravého okna vidíme nadcházející události (3) – např. naplánované události směrovacího protokolu, nebo události, které jsme nakonfigurovali dříve v scenarioManageru. V levém okně pak vidíme grafické zobrazení sítě a aktuální dění (4).

Na jednotlivé prvky v síti můžeme poklepat levým tlačítkem myši a zobrazit tak jejich parametry a vlastnosti. Jestliže třeba chceme zobrazit směrovací tabulku směrovače R1, poklepeme myší na směrovač, zobrazí se nám jeho vnitřní struktura, kde tentokrát poklepeme na `routing table` a následně na vektor `Routes`. Takto jsme se proklikali do směrovací tabulky, kde vidíme (zatím) pouze záznamy o přímo připojených sítích (Obrázek 3.18). Jakmile spustíme simulaci tlačítkem RUN, po nějaké době dojde ke konvergenci sítě a směrovací tabulky, stejně tak jako další parametry sítě, se změní.



Obrázek 3.18: *Směrovací tabulka R1*

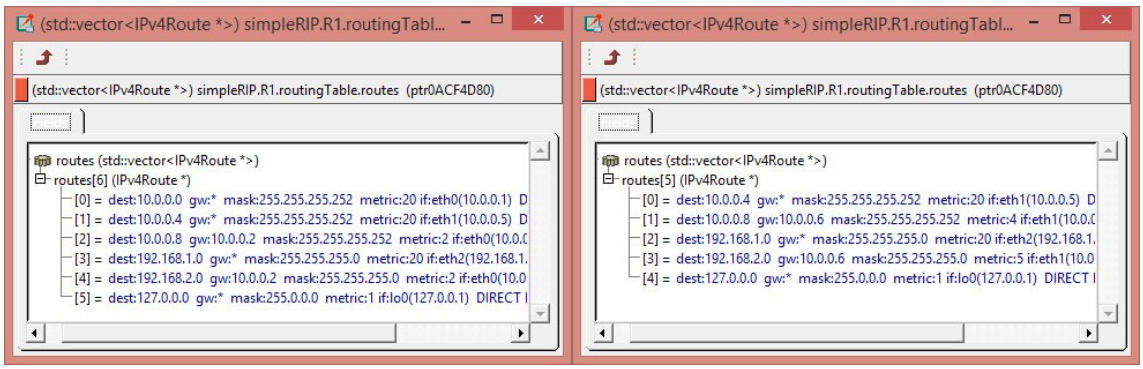
Nyní spustíme simulaci a po nějaké chvíli ji zase zastavíme, abychom se podívali, co se v síti událo. Na obrázku 3.19 je zachycený síťový provoz, který jsme vytvářeli v ini souboru. Můžeme vidět, že v čase T=4,2 počítače vysílají UDP pakety a hned potom se provádí enkapsulace.

```
** Event #466  T=4.2  simpleRIP.H1.networkLayer.ip (IPv4, id=108), on `UDPBasicAppData-2' (UDPPacket, id=283)
Sending datagram `UDPBasicAppData-2' with dest=192.168.2.2
Routing datagram `UDPBasicAppData-2' with dest=192.168.2.2: output interface is eth0, next-hop address: 192.168.1.1
Sending out packet to interface eth0
** Event #467  T=4.2  simpleRIP.H2.networkLayer.ip (IPv4, id=126), on `UDPBasicAppData-2' (UDPPacket, id=284)
Sending datagram `UDPBasicAppData-2' with dest=192.168.1.2
Routing datagram `UDPBasicAppData-2' with dest=192.168.1.2: output interface is eth0, next-hop address: 192.168.2.1
Sending out packet to interface eth0
** Event #468  T=4.2  simpleRIP.H1.eth[0].encap (EtherEncap, id=115), on `UDPBasicAppData-2' (IPv4Datagram, id=285)
Encapsulating higher layer packet `UDPBasicAppData-2' for MAC
```

Obrázek 3.19: *Zachycení UDP komunikace*

Další událost, kterou si rozebereme, je naše plánované odstavení linky. Najdeme si tedy událost v okně s událostmi – `scenario-event, at T=100`, klikneme pravým tlačítkem a zvolíme možnost „Express run until message“. Tím se simulace znovu spustí a zastaví se těsně před vykonáním zvolené události. Podíváme se, co obsahuje směrovací tabulka směrovače R1.

Jestliže se v simulaci posuneme o jeden krok, dojde k odpojení linky a tím přestanou být některé cesty platné. Směrovače tento fakt zaregistrují a začnou aktualizovat své tabulky. Posuneme tedy simulaci o nějaký čas vpřed a znovu se podíváme na směrovací tabulku. Tabulky před přerušením a po přerušení linky jsou na obrázku 3.20.



Obrázek 3.20: *Směrovací tabulka směrovače R1 před a po odpojení linky*

Náš projekt můžeme ještě rozšířit o více síťových zařízení, nebo můžeme pozměnit nastavení metriky protokolu RIP u jednotlivých linek a sledovat změnu chování směrovacího protokolu.

Závěr

Cílem mé bakalářské práce bylo navrhnout a popsat aplikaci směrovacích protokolů RIP, OSPF a BGP v simulátoru OMNeT++. V době tvorby této práce (akademický rok 2013/2014) byla aktuální verze simulátoru OMNeT++ 4.4.1 a modul INET byl dostupný ve verzi 2.3.0. Kvůli podpoře protokolu RIP jsem použil právě INET ve verzi 2.3.0. V nižších verzích nebyl RIP podporován.

V úvodní kapitole této práce jsem se zabýval teoretický popisem směrovacích protokolů – jejich rozdělením do skupin a popisem každého zástupce.

V druhé kapitole jsem se věnoval samotnému simulačnímu prostředí OMNeT++, jeho instalaci a také instalaci modulu INET, který jsem následně používal pro praktickou část této bakalářské práce.

Třetí kapitola byla věnována samotným simulacím směrovacích protokolů. Byly vytvořeny tři zadání – každé z nich pro jiný směrovací protokol. Každé zadání bylo krok po kroku popsáno od samotného vytvoření nového projektu, až po simulaci a sledování chování jednotlivých síťových protokolů a síťových prvků.

Při vytváření této práce, nebylo moc dostupných návodů na vytváření simulací. Proto je mým přínosem právě zdokumentování postupu tvorby základních projektů v OMNeTu. Na základě této práce by studenti měli být schopni pochopit strukturu a práci v simulátoru. Práce by také mohla tvořit základ pro simulace složitějších síťových topologií. Vytvořené ukázky by se daly rozšířit o další síťové protokoly, nebo třeba o další síťové prvky a události v síti.

Použitá literatura

[1] SPORTACK, Mark A. *Směrování v sítích IP*. Vyd. 1. Brno: Computer Press, 2004, 351 s. ISBN 80-251-0127-4.

[2] PUŽMANOVÁ, Rita. *TCP/IP v kostce*. 2. upr. a rozš. vyd. České Budějovice: Kopp, 2009, 619 s. ISBN 978-80-7232-388-3.

[3] TCP/IP směrování [online]. Dostupné z: http://www.samuraj-cz.com/clanek/tcpip-routing-smerovani/

[4] RFC 2080: RIPng for IPv6. [online]. Dostupné z: https://tools.ietf.org/html/rfc2080

[5] Routing Protocols Clasification [online]. Dostupné z: http://www.cs.vsb.cz/grygarek/SPS/lect/PREZENTACE/DV-routing-PG.pdf

[6] Směrovací protokol BGP [online]. Dostupné z: http://www.cs.vsb.cz/grygarek/SPS/lect/BGP/BGP.html

[7] SOSINSKY, Barrie. *Mistrovství – počítačové sítě*. Vyd. 1. Brno: Computer Press, 2010, 840 s. Mistrovství (Computer Press). ISBN 978-80-251-3363-7.

[8] THOMAS, Thomas M. *Juniper Networks reference guide: JUNOS routing, configuration, and architecture*. Boston: Addison-Wesley, c2003, xxvii, 845 p. ISBN 02-017-7592-1.

[9] OMNeT++ instalation guide. [online]. Dostupné z: http://omnetpp.org/doc/omnetpp/InstallGuide.pdf

[10] INET Framework Download Page. [online]. Dostupné z: http://inet.omnetpp.org/index.php?n=Main.Download

Seznam příloh

K bakalářské práci je jako příloha přiloženo CD, na kterém se nachází tato práce ve formátu .pdf, zdrojové kódy k jednotlivým projektům z prostředí OMNeT++ a instalační balík OMNeT++ ve verzi 4.4.1 a modul INET ve verzi 2.3.0.

Adrešář	Obsah
doc	Bakalářská práce ve formátu .PDF a .DOCX
ospf	Zdrojové kódy simulace protokolu OSPF
bgp	Zdrojové kódy simulace protokolu BGP
rip	Zdrojové kódy simulace protokolu RIP
omnet	Instalační balík OMNeT++ 4.4.1 pro Windows
inet	Framework INET ve verzi 2.3.0
inet_fix	Záplata pro INET 2.3.0 pro Windows

Tabulka 1: Obsah CD